DISEÑO FÍSICO DE BBDD. LENGUAJE DE DEFINICIÓN DE DATOS

Contenido

1. INTRODUCCIÓN

En la fase de análisis se realiza el ERS (Especificación de Requisitos Software). A partir de ésta especificación de requisitos se realiza el diseño conceptual de una BD mediante el Modelo E/R.

También vimos cómo hacer el modelo lógico mediante el modelo relacional que se obtenía a partir del modelo E/R y aprendimos a comprobar que ese modelo estaba normalizado.

Siguiendo con el proceso de desarrollo, lo que debemos hacer ahora es pasar al diseño físico de la BD. Es decir, implementar la Base de Datos. Para ello, se programarán las distintas tablas que constituirán la Base de datos. Todo esto se hará programando en el lenguaje más extendido para la definición y manipulación de datos en SGBDR: SQL.

2. MOTORES DE ALMACENAMIENTO EN MySQL

MySQL soporta diferentes tipos de almacenamiento de tablas (motores de almacenamiento o motores de almacenamiento, en inglés). Y cuando se crea una tabla es necesario especificar en qué sistema se posibles queremos crear.

Por defecto, MySQL a partir de la versión 5.5.5 crea las tablas de tipo InnoDB, que es un sistema transaccional, es decir, que soporta las características que hacen que una base de datos pueda garantizar que los datos se mantendrán consistentes.

Las propiedades que garantizan los sistemas transaccionales son las características llamadas ACID. ACID es el acrónimo inglés de atomicity, consistency, isolation, Durability:

Atomicidad: se dice que un SGBD garantiza atomicidad si cualquier transacción o bien finaliza correctamente (commit), o bien no deja rastro alguno de su ejecución (rollback).

Consistencia: se habla de consistencia cuando la concurrencia de distintas transacciones no puede producir resultados anómalos.

Aislamiento (o aislamiento): cada transacción dentro del sistema debe ejecutarse como si fuera la única que se ejecuta en ese momento.

Definitividad: si se confirma una transacción, en un SGBD, el resultado de ésta debe ser definitivo y no puede perderse.

Sólo el motor InnoDB permite crear un sistema transaccional en MySQL. Los otros tipos de almacenamiento no son transaccionales y no ofrecen control de integridad en las bases de datos creades.

Evidentemente, este sistema (InnoDB) de almacenamiento es el que a menudo interesará utilizar para las bases de datos que creamos, pero puede haber casos en los que sea interesante considerar otros tipos de motores de almacenamiento. Por eso, MySQL también ofrece otros sistemas como, por ejemplo:

MyISAM: era el sistema por defecto antes de la versión 5.5.5 de MySQL. Se utiliza mucho en aplicaciones web y en aplicaciones de almacén de datos (datawarehousing).

Memory: este sistema almacena todo en memoria RAM y, por tanto, se utiliza para sistemas que requieran un acceso muy rápido a los datos.

Merge: agrupa tablas de tipo MyISAM para optimizar listas y búsquedas. Las tablas necesarias agrupar deben ser similares, es decir, deben tener el mismo número y tipos de columnas.

Para obtener una lista de los motores de almacenamiento soportados por la versión MySQL que tenga instal·lada, el comando SHOW ENGINES.

3. EL LENGUAJE SQL

3.1.

El nacimiento del lenguaje SQL data de 1970 cuando EF Codd publica su libro: «Un modelo de datos relacional para grandes bancos de datos compartidos». Ese libro dictaría las direcrices de las bases de datos relacionales. Apenas dos años después IBM (para quien trabajaba Codd) utiliza las directrices de Codd para crear el Standard English Query Language (Lenguaje Estándar Inglés para Consultas) al que se le llamó SEQUEL. Más adelante se le asignaron las siglas SQL (Standard Query Language, lenguaje estándar de consulta) aunque en inglés se siguen pronunciando secuel. En español se pronuncia esecuele.

En 1979 Oracle presenta la primera implementación comercial del lenguaje. Poco después se convertía en un estándar en el mundo de las bases de datos avalado por los organismos ISO y ANSI. En 1986 se toma como lenguaje estándar para ANSI de los SGBD relacionales. Un año después la adopta ISO, lo que convierte a SQL en estándar mundial como lenguaje de bases de datos relacionados.

En 1989 aparece el estándar ISO (y ANSI) llamado SQL89 o SQL1. En 1992 aparece la nueva versión estándar de SQL (hoy en día sigue siendo la más conocida) llamada SQL92. se aprueba un nuevo SQL estándar que incorpora mejoras que incluyen triggers, procedimientos, funciones,… y otras características de las bases de datos objeto-relacionales; este estándar se conoce como SQL99. El último estándar es el del año 2011 (SQL2011) Elementos de SQL

SQL se basa en la Teoría Matemática del Álgebra Relacional.
varios elementos:

- Lenguaje de definición de datos (DDL): proporciona órdenes para definir, modificar o eliminar los distintos objetos de la base de datos (tablas, vistas, índices…).
- Lenguaje de Manipulación de Datos (DML): proporciona órdenes para insertar, suprimir y modificar registros o filas de las tablas. También contempla la realización de consultas sobre la BD.
- Lenguaje de Control de Datos (DCL): permite establecer derechos de acceso de los usuarios sobre los distintos objetos de la base de datos. Lo forman las instrucciones GRANT y REVOCAR.

- Oracle contempla además sentencias para transacciones. Administran las modificaciones creadas por las instrucciones DML. Lo forman las instrucciones ROLLBACK, COMMIT y PUNTO DE GUARDADO.

3.2. Proceso de ejecución de sentencia SQL

El proceso de una instrucción SQL es el siguiente:

1. Se analiza la instrucción. Para comprobar la sintaxis de ésta
2. Si es correcta se valora si los metadatos de la misma son correctos. en el diccionario de datos.
3. Si es correcta, se optimiza, a fin de consumir los mínimos recursos posibles.
4. Se ejecuta la sentencia y se muestra el resultado al emisor de ésta.

4. LENGUAJE DE DEFINICIÓN DE DATOS: DDL

4.1. Tipo de datos

Los tipos de datos que puede haber en un campo, se pueden agrupar en tres grandes grupos:

- Tipo de Fecha
- Tipos numéricos
- Tipo de Cadena

Puedes buscar información sobre los tipos de datos y sus características en MySQL 5.0 Reference Manual (Capítulo 11)

4.1.1 Tipo de Fecha

Tipo de Campo	Tamaño de Almacenamiento
FECHA	3 bytes
FECHA Y HORA	8 bytes
MARCA DE TIEMPO	4 bytes
TIEMPO	3 bytes
AÑO	1 byte

4.1.2. Tipos numéricos

Tipo de Campo	Tamaño de Almacenamiento
PEQUEÑO	1 byte
PEQUEÑO	2 bytes
MEDIUMINT	3 bytes
INT	4 bytes
ENTERO	4 bytes
GRANDE	8 bytes
FLOTANTE(X)	4 a 8 bytes
FLOTAR	4 bytes
DOBLE	8 bytes
DOBLE PRECISIÓN	8 bytes
REAL	8 bytes
DECIMALES(M,D)	M+2 bytes si D > 0, M+1 bytes si D = 0
NUMÉRICO(M,D)	M+2 bytes si D > 0, M+1 bytes si D = 0

Una columna puede tener el atributo adicional AUTO_INCREMENT (si el tipo es un INT). Cuando introducimos un valor de NULL (recomendado) o 0 en una columna AUTO_INCREMENT (autoindexada), la columna se asigna al siguiente valor de secuencia. Típicamente esto es value+1, donde value es el mayor valor posible para la columna en la mesa. Secuencias AUTO_INCREMENT comienzan con 1. Por ejemplo:

```
Prueba CREATE TABLE (un INT NOT NULL AUTO_INCREMENT)
```

4.1.3 Tipo de cadena

Tipo de campo	Tamaño de Almacenamiento
CARÁCTER(n)	n bytes
VARCHAR(n)	n +1 bytes
PEQUEÑOBLOB, PEQUEÑOTEXTO	Longitud+1byte
BLOQUE, TEXTO	Longitud +2 bytes
BLOB MEDIANO, TEXTO MEDIANO	Longitud +3 bytes
TEXTO LARGO, TEXTO LARGO	Longitud +4 bytes
ENUM('valor1','valor2',...)	1 o dos bytes dependiendo del número de valores

SET('valor1','valor2',...)	1, 2, 3, 4 o 8 bytes, depenent del nombre de valores

4.2 Creación de una Base de Datos

MySQL permite crear diferentes bases de datos en un mismo servidor (también se pueden denominar Schemas. Veamos cómo se crea una BD en MySQL.

```
CREAR {BASE DE DATOS | ESQUEMA} [SI NO EXISTE] nombre_base_de_datos

 [especificación_create [, especificación_create] …]

especificación_create:[DEFAULT] CHARACTER SET juego_de_caracteres
 |[DEFAULT] COLLATE nombre_de_colación
```

El comando es CREATE, después se puede especificar DATABASE o SCHEMA, es indiferente utilizar cualquiera de las dos para crear la base de datos.

Cómo leer la sintaxis:

- Cuando se pueden elegir un elemento entre varios éstos van entre claves { } y separados por uno tubo |.
- Por otra parte, los elementos que van entre corchetes [] son opcionales, pueden ir o no, por ejemplo es opcional poner IF NOT EXIST, es decir: que cree la base de datos si no existe ya. Por último, puedes observar la especificación del conjunto de caracteres a utilizar o bien el modo tratar ese conjunto de caracteres. Aunque, como se ha dicho, no es necesario y normalmente no especificaremos estos parámetros

Por ejemplo, si utilizamos CHARACTER SET utf8, estamos diciendo que se utilice utf8 para almacenar los caracteres. Si indicamos COLLATE uft8_spanish_ci significa que se utilizará el alfabeto español para las ordenaciones y comparaciones en nuestra base de datos. importante para que al ordenar alfabéticamente la ñ y las letras acentuadas vayan en su lugar y para que al comparar datos acentuados y otros no acentuados sean considerados iguales (é es igual a e).

En un Query del WorkBench pondremos:

```
CREAR BASE DE DATOS prueba1 CONJUNTO DE CARACTERES utf8 COLLATE utf8_spanish_ci;

Consulta correcta, 1 fila afectada (0,00 seg)
```

MySQL nos indica que el comando se ha ejecutado correctamente (Query OK).

Si intentamos crear de nuevo la misma base de datos se producirá un error. Nos indica que no se ha creado la base de datos porque ya existe.

Si queremos evitar este error debemos utilizar la cláusula opcional IF NOT EXISTS.

```
CREAR BASE DE DATOS SI NO EXISTE prueba1 CONJUNTO DE CARACTERES utf8 COLLATE
utf8_español_ci;

Consulta correcta, 1 fila afectada, 1 advertencia (0,00 seg)
```

En este caso la ejecución ha sido correcta y nos muestra un aviso (1 warning) indicando que no se ha creado la base de datos.

Cada vez que tengamos una BD desde el punto de vista lógico (simplificando: una serie de tablas relacionadas entre sí) crearemos un SCHEMA para agruparlas. Así, todos los objetos de BD que pertenecen a una misma BD lógica deben estar en el mismo SCHEMA.

4.3. Creación, Modificación y Eliminación de tablas

En este apartado veremos los comandos SQL que se utilizarán para crear y modificar la definición de una tabla, así como para eliminarla de la base de datos.

Nos referiremos al sistema MySQL; pudiendo haber diferencias respecto a otros SGBD.

4.3.1. Creación de Tablas

El nombre de las tablas debe cumplir las siguientes reglas:

- Deben empezar con una letra
- No deben tener más de 30 caracteres
- Sólo se permiten utilizar letras del alfabeto (inglés), números o el signo de subrayado (también el signo $ y #, pero esos se utilizan de manera especial por lo que no son recomendados)
- No puede haber dos tablas con el mismo nombre para el mismo usuario (pueden coincidir los nombres si están en diferentes esquemas)
- No puede coincidir con el nombre de una palabra reservada de SQL

Para la creación de tablas con SQL se utiliza el comando CREATE TABLE. Este comando tiene una sintaxis más compleja de la que aquí se expone, pero empezaremos por la sintaxis básica. Sintaxis básica de creación de tablas:

```
CREAR TABLA nom_taula [SI NO EXISTE] (
  columna1 tipo [ restricciones de columna1 ],
  columna2 tipo [ restricciones de columna2 ],
  tipo columna3 [ restricciones columna3 ],
  ...
  [ restricciones de mesa ]
);
```

Para realizar las separaciones se utiliza la coma. La última línea, antes del paréntesis de cierre, no lleva coma.

Donde las restricciones de columna tienen la siguiente sintaxis:

```
{[NOT] NULL | UNIQUE | PRIMARY KEY | DEFAULT valor | CHECK (condició) |
 REFERENCES nombre_tabla (nombre_columna)[ON DELETE {SET NULL|CASCADE}]
[AL ACTUALIZAR {ESTABLECER NULL|CASCADE}]}
```

Nota: La cláusula CHECK se analiza pero se ignora en todos los motores de almacenamiento MySql.

Y las restricciones de tabla tienen la siguiente sintaxis:

```
[CONSTRAINT [nombre_restricción]] {
    CLAVE PRIMARIA (columna1 [,columna2] ... )
| ÚNICO (columna1 [,columna2] ... )
| CLAVE EXTERNA (columna1 [,columna2] ... ) REFERENCIAS nom_taula
(columna1 [,columna2] ... ) [ON DELETE {CASCADE | SET NULL}][ON UPDATE
{CASCADA | ESTABLECER NULO}]}
| CHECK (condición)}
```

Obligatoriamente debemos crear una restricción de mesa cuando una misma restricción afecte varias columnas. Por ejemplo, si tenemos una clave primaria compuesta por varios campos, hemos de establecer una restricción de tabla, no de columna.

El significado de las diferentes opciones que aparecen en la sintaxis CREATE TABLE es:

- PRIMARY KEY: establece ese atributo o conjunto de atributos como la clave primaria de la mesa. Esta restricción implica ya las restricciones UNIQUE y NOT NULL.
- UNIQUE: impide que se introduzcan valores repetidos para ese atributo o conjunto de atributos. No se puede utilizar junto con PRIMARY KEY. alternativas.
- NOT NULL: evita que se introduzcan filas en la tabla con valor NULL para ese atributo. No se utiliza con PRIMARY KEY.
- DEFAULT valor_por_defecto: permite asignar un valor por defecto al campo que se está definiendo. El valor no puede ser devuelto por una función. Por ejemplo, no sería correcte:

```
Fecha DataNaix PREDETERMINADA CURDATE()

incorrecte
```

Lo correcto sería usar la constante CURRENT_TIMESTAMP

```
Marca de tiempo actual predeterminada de DataNaix
```

- CHECK (condición): permite establecer condiciones que deben cumplir los valores de la tabla que se introducirán en esta columna.

  Nota: La cláusula CHECK se analiza pero se ignora en todos los motores de almacenamiento

- FOREIGN KEY: define una clave externa (ajena) de la tabla respecto a otra tabla. Esta restricción especifica una columna o una lista de columnas como clave externa de una tabla referenciada. No puede definirse una restricción de integridad referencial que se refiere a una tabla antes de que esta tabla haya sido creada. Es importante resaltar que una clave externa debe referenciar a una clave primaria completa de la tabla padre, y nunca en un subconjunto de los atributos que forman esta clave primaria.

  DIRECTRICES DE BORRADO O MODIFICACIÓN

  Al definir una clave ajena, podemos definir una política de Borrado y una de Modificación. Es decir, podemos definir qué va a pasar con los valores de las columnas que conforman la

clave externa en caso de que la fila a la que hacen referencia en la tabla principal se elimine o se elimine modifique su valor en la Clave Primaria.

LIBRO

id_lib	título	tipo	autor_id
LIB-000016	Crónica de una muerte anunciada	Novela	GAGA
LIB-000017	Siempre NO	Teatro	GAGA
LIB-000008	Doce cuentos peregrinos	Cuento	GAGA
LIB-000001	El club de los suicidas	Novela	ROST
LIB-000004	Poemas	Poesía	BERU

AUTOR

autor_id	nombre
GAGA	Gálamo Gante
ROST	Robert Steinball
BERU	Bertrand Rusbelt

En la mesa LIBRO está definida una Llave Aliena hacia la mesa AUTOR. ¿Qué pasa con ¿'LIB-000016', 'LIB-000017' y 'LIB-000008' si se eliminan en la tabla AUTOR a 'GAGA'?

¿Qué sucede con 'LIB-000016', 'LIB-000017' y 'LIB-000008' si se modifica en la tabla? AUTOR el camp autor_id ¿'GAGA' se convierte en 'GáGa'?

- o WHERE DELETE CASCADE: especifica que el integridad referencial borrando los valores de la clave externa correspondientes a un valor borrado de la tabla referenciada (tabla padre). Si se omite esta opción no se permitirá borrar valores de una tabla que sean referenciados como clave externa en otras tablas.
- o DONDE DELETE SET NULL: especifica que se ponga a NULL los valores de la clave externo correspondiente a un valor eliminado de la tabla referenciada (tabla principal).
- o DONDE DELETE NO ACTION. Opción por defecto. Un intento de borrar una fila o actualizar un valor de clave primaria no se permitirá si la fila está referenciada. RESTRICT seria equivalente en MySQL

Usar

El valor predeterminado es que las filas de la tabla principal no se pueden eliminar si existe una fila en la tabla secundaria que se refiere a esta fila principal, si no indicamos DONDE DELETE CASCADE o DONDE DELETE SIETE NULL. El estándar SQL define muchas opciones.

El estándar SQL define 5 opciones para manejar esta situación de tablas principal/secundaria de diversas formas. Estas opciones son:

- WHERE DELETE CASCADE: si se elimina una fila de la tabla principal, todos los filas coincidentes en la tabla secundaria.
- DÓNDE DELETE SET NULL: si se elimina una fila de la tabla principal, todas las columnas de referencia en todas las filas coincidentes de la tabla secundaria se establecen en NULL.
- DÓNDE DELETE SET DEFAULT: si se elimina una fila de la tabla principal, todas las columnas de referencia en todas las filas coincidentes de la tabla secundaria se configuran en el valor predeterminado de la columna.
- DÓNDE DELETE RESTRICTO: está prohibido eliminar una fila de la tabla principal si esa fila tiene alguna fila coincidente en la tabla secundaria. El punto en el tiempo cuando realiza la comprobación se puede aplazar hasta que se realice CONMIT.
- ON DELETE NO ACTION (el valor predeterminado): se prohíbe eliminar una fila de la tabla primaria si esa fila tiene filas coincidentes en la tabla secundaria.

Análoga a la opción ON DELETE existe una opción ON UPDATE Define las mismas 5 opciones para al caso de cambiar una columna en la tabla principal a la que hace referencia la columna de una tabla secundaria.

- DÓNDE UPDATE CASCADE: Cualquier cambio en una columna referenciada en la tabla primaria provoca el mismo cambio en la columna de referencia correspondiente en las filas coincidentes de la mesa secundaria.
- DONDE UPDATE SET NULL: Cualquier cambio en una columna referenciada en la tabla primaria provoca que la columna de referencia correspondiente en las filas coincidentes de la tabla secundaria se establezca como nula.
- AL ACTUALIZAR ESTABLECER POR DEFECTO: cualquier cambio en una columna referenciada en la tabla principal provoca que la columna de referencia correspondiente en las filas coincidentes de la tabla de secundaria se establezca en su valor predeterminado.
- DÓNDE UPDATE RESTRICT: está prohibido cambiar una fila de la tabla principal si esa fila tiene filas coincidentes en la tabla secundaria. El punto en el tiempo cuando se realiza la comprobación se puede aplazar hasta que se realice CONMIT.
- DONDE UPDATE NO ACTION (valor predeterminado): está prohibido cambiar una fila de la tabla principal si esa fila tiene alguna fila coincidente en la tabla secundaria.

Si ON DELETE o ON UPDATE no están especificados, se producirá la acción predeterminada NO ACTION. En algunos sistemas, NO ACTION se implementa en el sentido de la opción RESTRICT. Cómo voces NO hay diferencia a efectos prácticos entre una y otra opción.

En la definición de una tabla pueden aparecer varias cláusulas FOREIGN KEY, tantas como claves externas tenga la mesa. Sin embargo, sólo puede existir una clave primaria, si bien esta clave primaria puede estar formada por varios atributos.

Ejemplos:

```
CREAR TABLA usuarios (
  Id CLAVE PRIMARIA ENTERA,
  dni CHAR(9) UNIQUE,
  nombre VARCHAR(50) NO NULO,
  edat INTEGER CHECK (edat>=0 and edat<120)
);
```

En caso de que queramos asignar un nombre a las restricciones tendremos que definirlas después de definir los campos. Así, si deseamos modificar posteriormente el diseño de la mesa, será más fácil gestionar las restricciones.

Criterios de notación para los nombres de restricciones (Aconsejables)

Para la Restricción de Clave principal (sólo una en cada mesa):

```
RESTRICCIÓN taula_camp_pk CLAVE PRIMARIA ...
```

Para Restricciones de Llave foránea (puede haber varias en cada mesa):

```
RESTRICCIÓN taula_camp_fk1 CLAVE EXTRAÑA ...
RESTRICCIÓN taula_camp_fk2 CLAVE EXTRAÑA ...
RESTRICCIÓN taula_camp_fk3 CLAVE EXTRAÑA ...
```

Para Restricciones de tipo UNIQUE (puede haber varias en cada mesa)

```
RESTRICCIÓN taula_camp_uq1 ÚNICA ...
RESTRICCIÓN taula_camp_uq2 ÚNICA ...
...
```

Ejemplo:

```
CREAR TABLA COTXES (
  matricula VARCHAR(8),
  marca VARCHAR(15) NO NULL,
  colorVARCHAR(15),
  bacalaoTaller VARCHAR(10),
  codPropVARCHAR(10),
  CONSTRAINT cotxes_mat_pk PRIMARY KEY (matricula),
  RESTRICCIÓN cotxes_codtaller_fk1 CLAVE EXTERNA (codTaller)
  REFERENCIAS TALLER(codTaller)
  AL ACTUALIZAR EN CASCADA AL ELIMINAR ESTABLECER NULL,
  RESTRICCIÓN cotxes_codprop_fk2 CLAVE EXTERNA (codProp)
  REFERENCIAS PROPIETARIO(codProp)
  EN CASCADA DE ACTUALIZACIÓN EN CASCADA DE ELIMINACIÓN,
  CONSTRAINT coches_color_ck1
  VERIFICAR (color EN ('ROJO','AZUL','BLANCO','GRIS','VERDE','NEGRO'))
);
```

En el ejemplo, las directrices de borrado y modificación definidas sobre las 2 claves ajenas determinarien que:

Si se MODIFICARA en la tabla TALLER el campo CodTaller, se modificarían automáticamente en CASCADA el campo CodTaller de la mesa COCHES de las filas relacionadas con ese taller.

Si BORRARAS en la tabla TALLER una fila, las filas de la tabla COCHES relacionadas con ésta pasarían a tener en su campo CodTaller el valor NULL.

Si se MODIFICARA en la tabla PROPIETARIO el campo CodProp, se modificarían automáticamente en CASCADA el campo CodProp de la mesa COCHES de las filas relacionadas con ese propietario.

Si BORRARAS en la tabla PROPIETARIO una fila, las filas de la tabla COCHES relacionadas se borrarían.

4.3.2. Eliminación de Tablas

La sentencia en SQL para eliminar tablas es DROP TABLE.

```
TABLA DROP nombre_taula
 [ CASCADA|RESTRINGIR];
```

RESTRICT y CASCADE se permiten para realizar la portabilidad más fácil. Por el momento, no hacen nada.

El borrado de una tabla es irreversible y no existe ninguna petición de confirmación, por lo que conviene ser muy cuidadoso con esta operación. Al borrar una tabla se borran todos los datos que conté.

Ejemplos:

```
CUNAS DE MESA ABATIBLES ;
```

Se eliminará la mesa COCHES, siempre que se pueda: no lo impidan las reglas de integridad.
Por ejemplo: no sea COCHE una mesa hacia la que están dirigidas Llaves Ajenas en otras tablas.

4.3.3. Modificación de Tablas

Cambiar de nombre una tabla

El mandato RENAME permite el cambio de nombre de cualquier objeto. Sintaxis:

```
RENOMBRAR nom A nom_nou;
```

Ejemplo:

```
RENAME COTXES TO AUTOMOVILES;
```

Cambia el nombre de la mesa COCHES ya partir de ese momento se llamará AUTOMOVILES

Borrar el contenido de una tabla

El mandato TRUNCATE TABLE seguida del nombre de una tabla, hace que se elimine el contenido de la tabla, pero no la mesa en sí . Incluso borra del archivo de datos el espacio ocupado por la mesa.
orden no puede anularse con un ROLLBACK)

Ejemplo:

```
TRUNCATE TABLE AUTOMOVILES;
```

Borra los datos de la tabla AUTOMOVILES.

4.3.3.1 Trabajo con columnas y restricciones

La cláusula ALTER TABLE permite realizar cambios en la estructura de una tabla: añadir columna, borrar columna, modificar columna.

Añadir Columnas

```
ALTERAR TABLA nombre AGREGAR (
  columna1 tipo [ restricciones ][,
  columna2 tipo [ restricciones ]
  ... ]
);
```

Permite añadir nuevas columnas a la tabla. Se deben indicar su tipo de datos y sus tipos propiedades si es necesario (al estilo de CREATE TABLE). Las nuevas columnas se añaden al final, no puede indicarse otra posición.

Ejemplos:

Añadimos la columna “fechaMatric” a la tabla VEHÍCULOS:

```
ALTERAR TABLA VEHÍCULOS AÑADIR ( closeMatric Date );
```

Añadimos las columnas “fechaMatric” y “tipoFaros” a la tabla VEHÍCULOS:

```
ALTERAR TABLA VEHICULOS AGREGAR (
  cerrar fecha matricial,
  tipoFaros VARCHAR(20) NO NULO
);
```

Borrar Columnas

```
ALTER TABLE nombre_taula DROP (nombre_columna, nombre_columna2, ...);
```

Elimina la columna indicada de forma irreversible e incluyendo los datos que contenía.

pueden eliminar todas las columnas, para la última columna será necesario usar DROP TABLE.

Ejemplo:

```
ALTER TABLE VEHICULOS DROP (tipoFaros);
```

Borra la columna “tipoFaros” de la tabla VEHICULOS y los datos que contuviera de forma irreversible.

Modificar columnas

Permite cambiar el tipo de datos y propiedades de una determinada columna. Sintaxis:

```
ALTERAR TABLA nombre_taula MODIFICAR (
  columna1 tipo [ restricciones de columna1 ]
 [, columna2 tipo [ restricciones de columna2 ] ... ]
);
```

Ejemplo:

```
ALTER TABLE AUTOMOVILES
MODIFICAR (color VARCHAR(20) NO NULO, codTaller VARCHAR(15));
```

Modifica dos campos o columnas de la tabla AUTOMOVILES cambiando su tamaño y además en Color, añadiendo la condición de que sea no nulo.

Los cambios que se permiten son:

- Incrementar precisión o anchura de los tipos de datos
- Sólo se puede reducir la anchura máxima de un campo si esa columna posee nulos en todos los registros, o no existen registros.
- Se puede pasar de CHAR a VARCHAR y viceversa (si no se modifica la anchura).

Añadir Restricciones

Sabemos que una restricción es una condición de obligado cumplimiento para una o más columnas de la mesa. A cada restricción se le pone un nombre, en el caso de no poner un nombre (en las que esto sea posible) entonces el propio le coloca el nombre que es un mnemotécnico con el nombre de tabla, columna y tipos de restricción.

Hemos visto que se pueden añadir al crear la tabla, o bien podemos hacerlo mediante modificación posterior de la taula.

También se puede modificar una restricción creada en . Su sintaxis general es:

```
ALTER TABLE nombre_tabla ADD Definición_Restricción [, Definición_Restricción…];
```

Definición_Restricción: La definición es la misma que cuando definimos una restricción en la creación de una mesa

Borrar Restricciones

Su sintaxis es la siguiente:

```
ALTERAR TABLA nombre_taula

DROP { CLAVE PRIMARIA | ÍNDICE nombre_índice | CLAVE EXTERNA símbolo_fk }
```

Si lo que queremos es redefinir (modificar) una restricción, lo que debemos hacer es:

Primero borrar la restricción y después volver a crearla de la forma deseada

Cambiar de nombre la Restricciones

Para ello se utiliza este comando:

```
ALTER TABLE nombre_tabla RENAME CONSTRAINT nombre_restricción TO
nombre_restricción_nuevo;
```

4.4. Creación, Modificación y Eliminación de vistas

Una vista no es más que una consulta almacenada a fin de utilizarla tantas veces como se deseo. Una vista no contiene datos sino la instrucción SELECT necesaria para crear la vista, esto asegura que los datos sean coherentes al utilizar los datos almacenados en las tablas.

Las vistas se utilizan para:

- Realizar consultas complejas más fácilmente
- Proporcionar tablas con datos completos
- Utilizar visiones especiales de los datos

Hay dos tipos de vistas:

- Simples. Las forman una sola tabla y no contienen funciones de agrupación. Su ventaja es que permiten siempre realizar operaciones DML sobre ellas.
- Complejas. Obtienen datos de varias tablas, pueden utilizar funciones de agrupación. No siempre permiten operaciones DML.

4.4.1. Creación de Vistas

Sintaxis:

```
CREAR [ O REEMPLAZAR ] VISTA nom_vista [ (alias1 [, alias2] ...) ]
COMO SELECCIONAR ...
```

- ORO REPLACE. Especifique ORO REPLACE para volver a crear la vista si ya existe. Puede utilizar esta cláusula para cambiar la definición de una vista existente sin eliminar, volver a crear y volver a conceder los privilegios de objeto previamente concedidos.
- sobrenombre. Lista de alias que se establecen para las columnas devueltas por la consulta SELECT en la que se basa esta vista. El número de sobrenombre debe coincidir con el

número de columnas devueltas por SELECT.

profundidad en el siguiente tema.

Lo bueno de las vistas es que después de su creación se utilizan como si fueran una mesa.
La vista USER_VIEWS del diccionario de datos permite mostrar una lista de todas las vistas que posee el usuario actual. La columna TEXTO de esa vista contiene la sentencia SQL que se va utilizar para crear la vista (sentencia que es ejecutada cada vez que se invoca a la vista).

4.4.2. Eliminación de Vistas

Se utiliza el comando DROP VIEW:

```
VISTA DEL GOLPE nombre_vista;
```

5. EL DICCIONARIO DE DATOS

El diccionario de datos es el lugar donde se deposita información sobre todos los datos que forman la BD: METADADOS. Es una guía en la que se describe la BD y los objetos que la forman. características lógicas de los sitios
donde se almacenan los datos del sistema, incluyendo nombre, descripción, alias, contenido y organización. Identifica los procesos donde se utilizan los datos y los lugares donde se necesita el acceso inmediato a la información.

El Diccionario de Datos (o catálogo de metadatos). es una de las partes más importantes de un SGBD El diccionario de datos proporciona información al DBA y al propio sistema de todas las bases de datos y tablas almacenadas. La información que proporciona el diccionario de datos puede ser desde la estructura de las tablas hasta privilegios de los usuarios o estado del consumo de los recursos.

Cada SGBD organiza su propio catálogo con diferentes estructuras y MySQL dispone de una base de datos especial llamada information schema donde se almacena la definición de las tablas, vistas, índices, estado del SGBD y otra llamada mysql donde almacenan los usuarios y privilegios.

Cuando veamos el tema de consultas, volveremos sobre este punto.

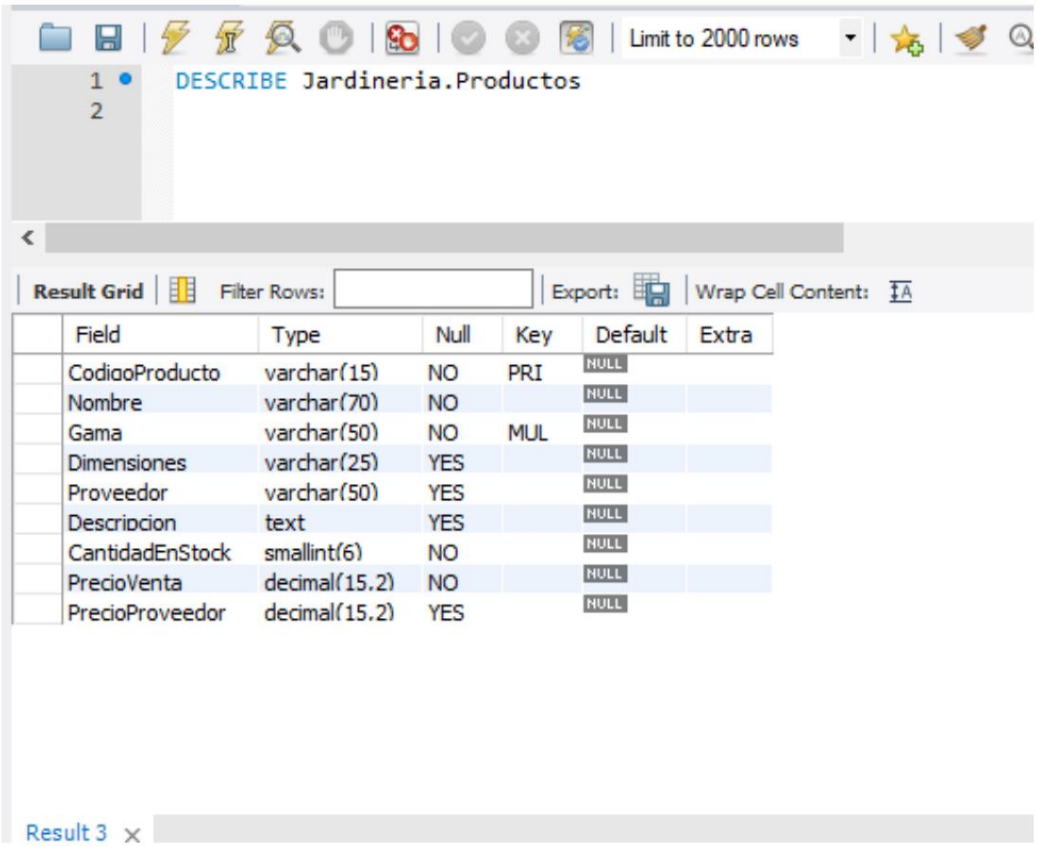
5.1 DESCRIBIR orden

El comando DESCRIBE, permite obtener la estructura de una mesa.

Ejemplo:

```
DESCRIBE Jardineria.Productos;
```

Aparecen los campos de la mesa 'Productos'. Esta instrucción no es parte del SQL estándar, pero casi es considerada así puesto que prácticamente todos los SGBD la utilizan.



Field	Type	Null	Key	Default	Extra
CodigoProducto	varchar(15)	NO	PRI	NULL	
Nombre	varchar(70)	NO		NULL	
Gama	varchar(50)	NO	MUL	NULL	
Dimensiones	varchar(25)	YES		NULL	
Proveedor	varchar(50)	YES		NULL	
Descripcion	text	YES		NULL	
CantidadEnStock	smallint(6)	NO		NULL	
PrecioVenta	decimal(15,2)	NO		NULL	
PrecioProveedor	decimal(15,2)	YES		NULL	